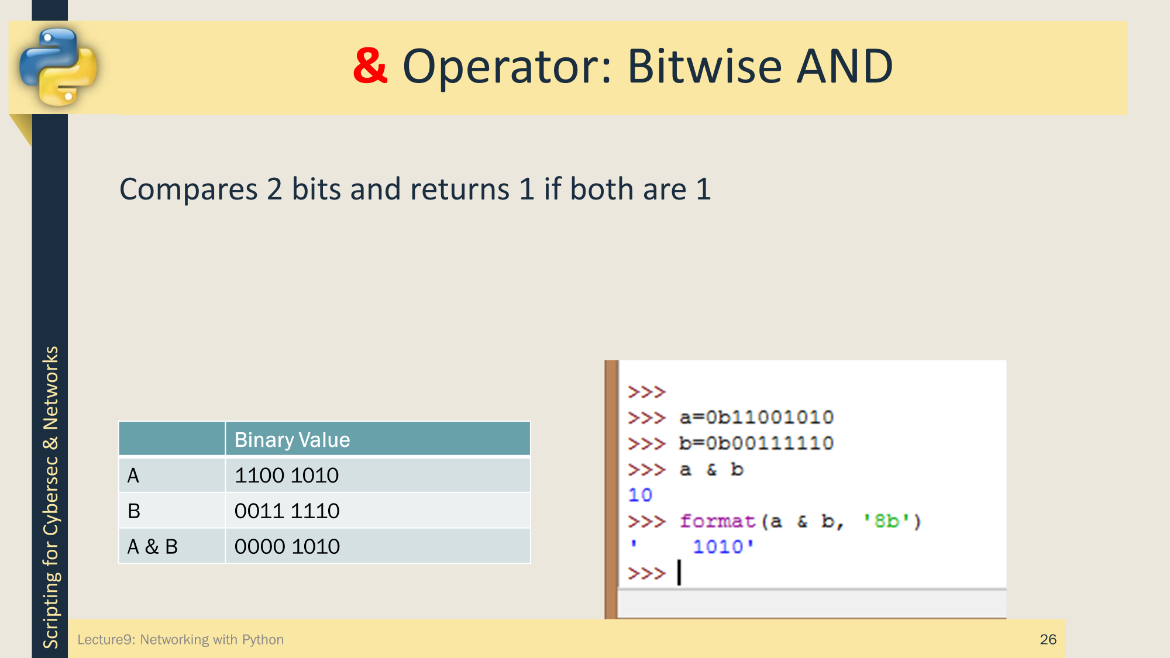
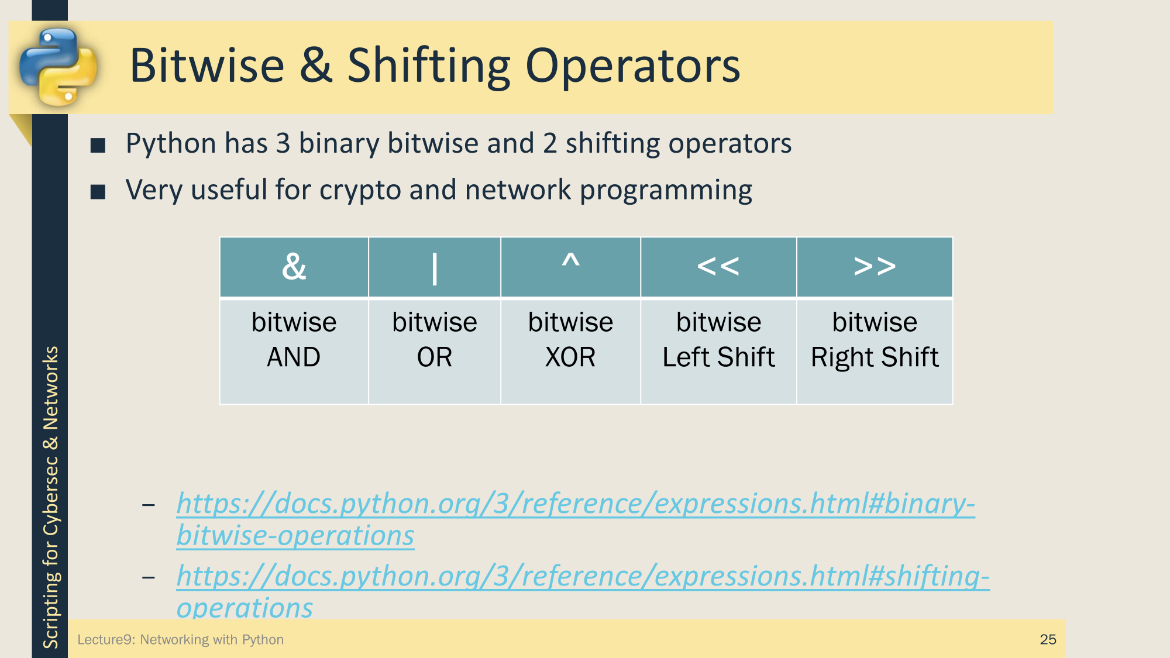
Lab 9 optional extension:

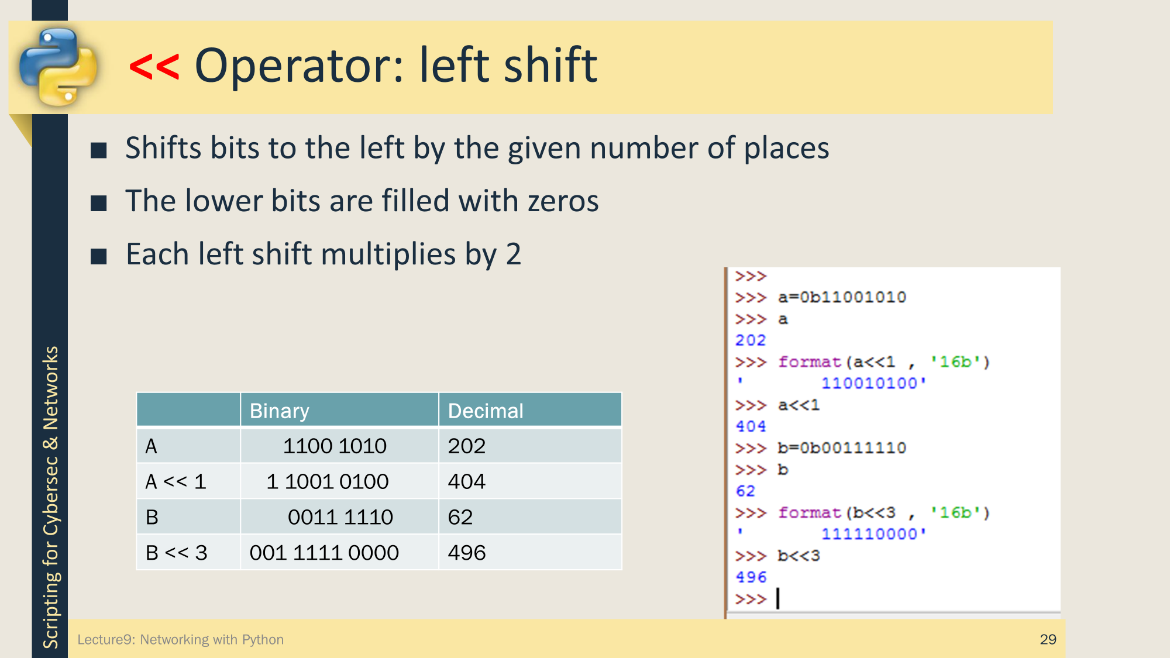
Manual decoding of IP addresses

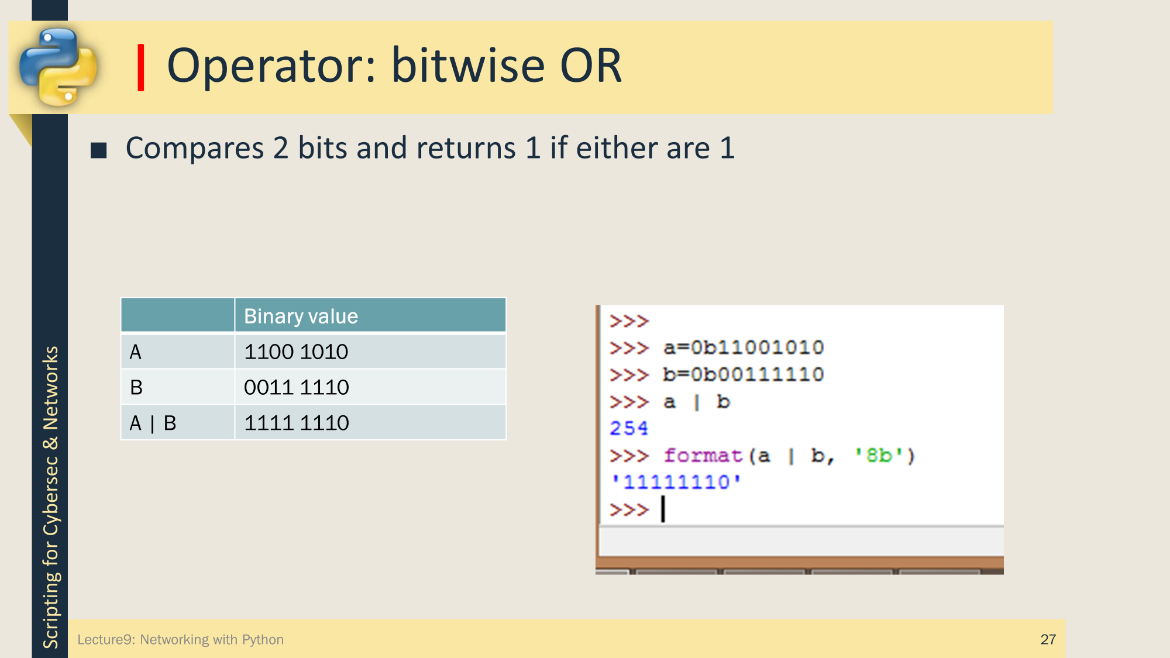
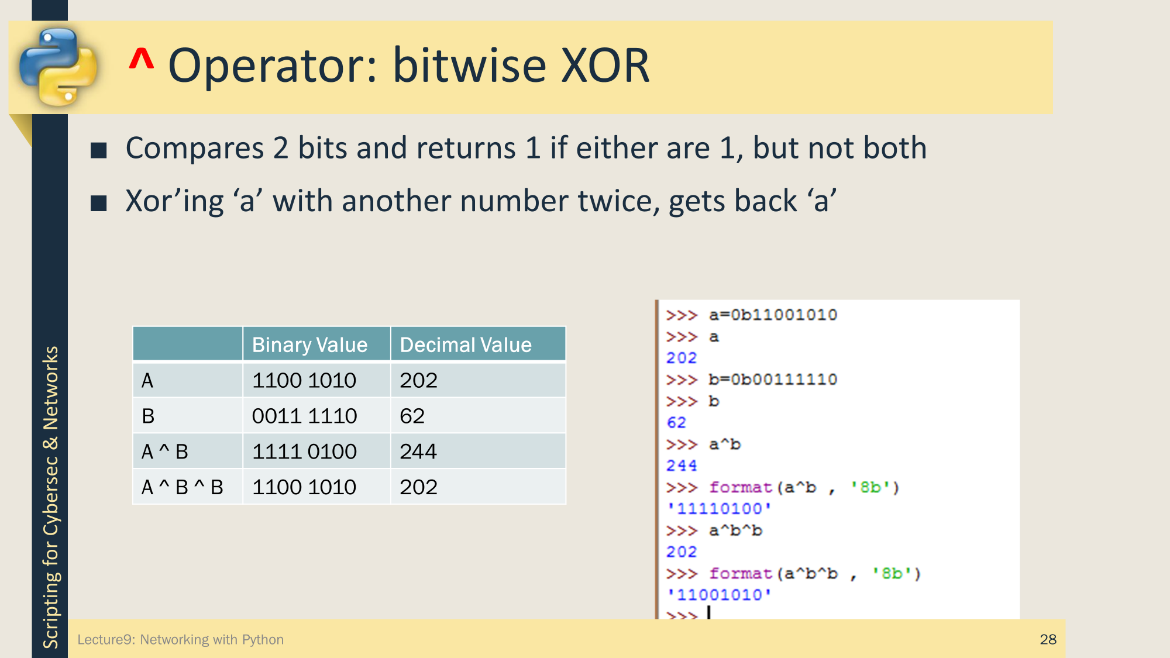
In the lab, Exercise 2.1.4, you used socket.inet\_ntoa() to decode the IP addresses from binary format.

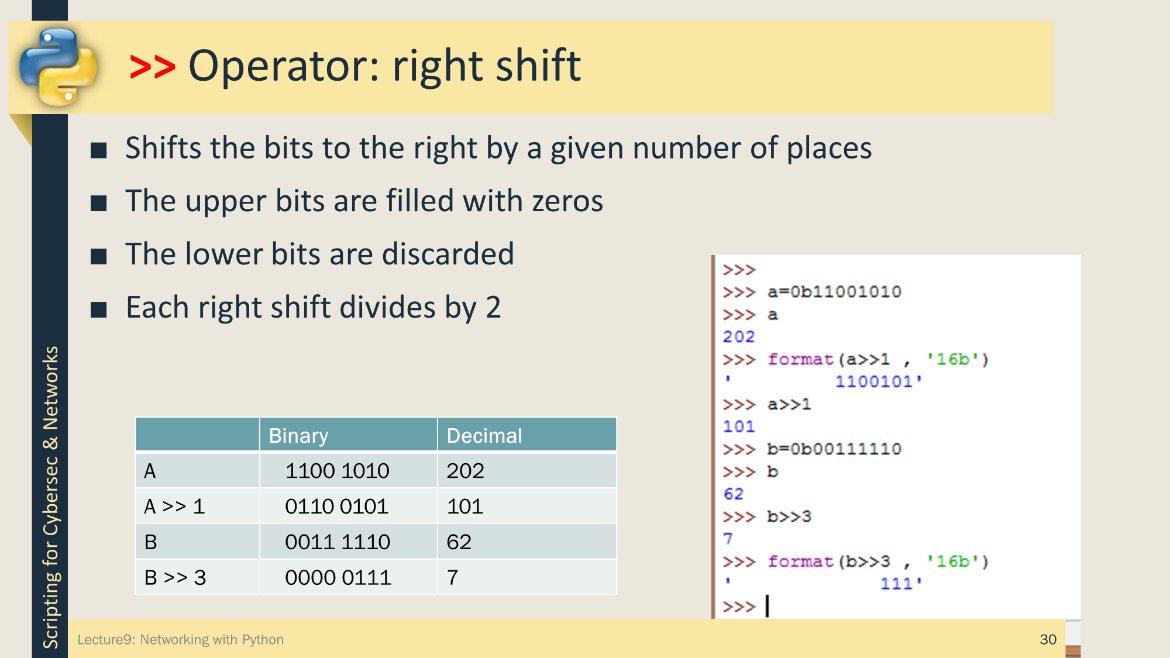
In this optional extension, you will learn from first principles how the process actually works, and write your own decoding function.

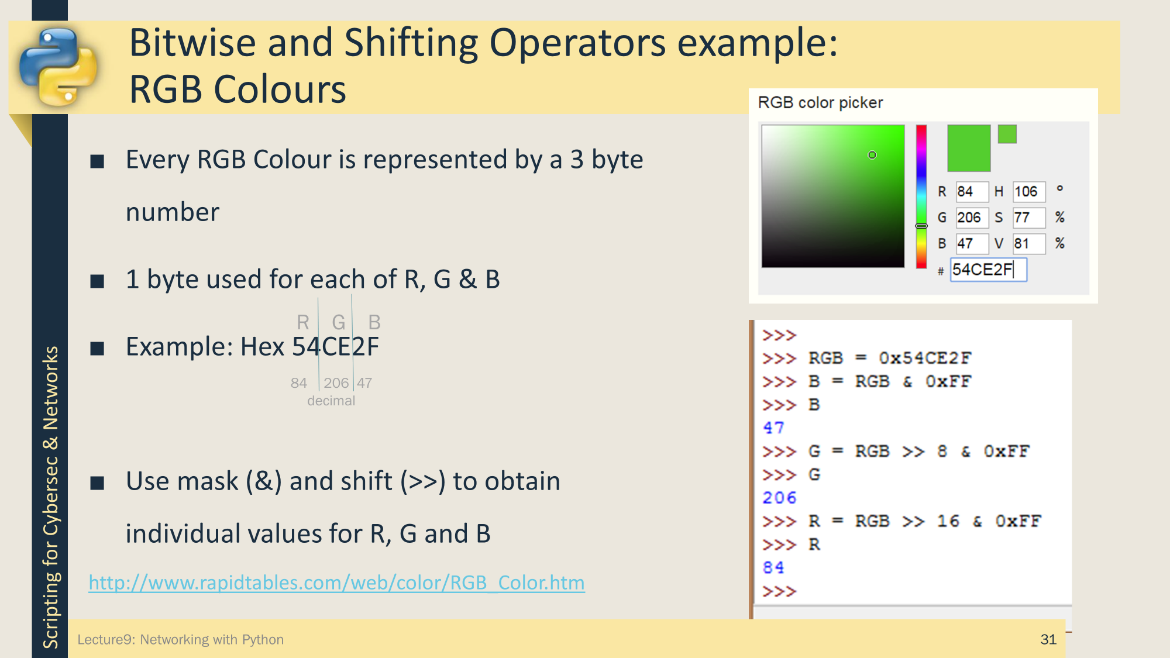
# Background information

The slides below introduce the bitwise and shifting operators that we will use in tis exercise.









Reformat IP addresses into standard dotted decimal format

ip addresses are generally shown in this format: ‘127.0.0.1’. But in the pcap record they’re a 32 bit binary streams e.g. *src=b'\x92\xb0\xa4[', dst=b'\x17\x156\x83'*. Each byte (8 bits) represents one of the octets in the dotted decimal format.

e.g b'\x92\xb0\xa4[' is 146.176.164.91

because: \x92 = 146 decimal; \xb0 = 176 decimal; \xa4 = 164 decimal and [ is 91 decimal.

Create a new function called decode\_ip\_v2() which will take an ip address as a binary stream and return the ip address in dotted decimal notation. Use the code below as a starter and refer to the lecture notes on Bitwise operations, specifically the example for RGB colours, for help. Remember to change your print statement in main() to call decode\_ip\_v2 for the src and dest ip addresses.

def decode\_ip\_v2(orig\_ip):

# convert orig\_ip to an integer

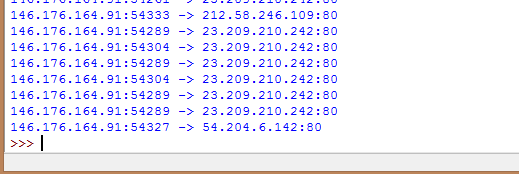
pass

# construct decoded\_ip as a dotted decimal string using '>>' and '&'

decoded\_ip = ''

return decoded\_ip

The output should now look similar to this….



Reformat IP address using a Loop or List Comprehension - Challenge Question

If you’ve not already done so, reformat the ip address using a loop. Once that’s done, rewrite the loop into a list comprehension. List comprehensions are difficult to become familiar with but can be more easily written if an intermediate looping step is taken. For example:

# Looping method

# build an array of octets then join them using ','.join()

octets = []

for i in Looping\_Criteria:

octets.append( Code\_to\_Generate\_Next\_Octet )

decoded\_ip = '.'.join(octets)

# List comprehension just rearranges these into 2 lines

octets = [ Code\_to\_Generate\_Next\_Octet for i in Looping\_Criteria ]

decoded\_ip = '.'.join(octets)

# Or all in 1 line!!

decoded\_ip = '.'.join( [ Code\_to\_Generate\_Next\_Octet for i in Looping\_Criteria ] )